Clustering for Traceability Managing in System Specifications

Manel Mezghani, Juyeon Kang Choi, Eun-Bee Kang, Florence Sèdes

Clustering for traceability managing in system specifications

Manel Mezghani
Semios for requirements
OneLight Studio
Toulouse, France
mezghani.manel@gmail.com

Juyeon Kang
Fortia Financial Solutions
Paris, France
juyeon.kang@fortia.fr

Eun-Bee Kang
Semios for requirements
OneLight Studio
Toulouse, France
e.kang@semiosapp.com

Florence Sèdes
IRIT, University of Toulouse
CNRS, INPT, UPS, UT1, UT2J,
France
florence.sedes@irit.fr

***Abstract*—System specifications are generally organized according to several documents hierarchies levels linked in order to represent the traceability information. Requirements engineering experts verify manually the links between each specification which allows to generate a traceability matrix. The purpose of this paper is to automatize the generation of the traceability matrix since it is a time consuming and costly task. We propose an artificial intelligence based approach to deal with this problem through a clustering approach. This latter is an unsupervised algorithm that doesn't need any prior knowledge on the language neither the domain of the specifications. Our approach generates duplicates and clusters containing linked requirements. We experiment our approach in an aeronautic domain and a space domain. We obtain better results for high level specifications especially with a pre-processing.**

***Index Terms*—Requirements engineering, traceability, clustering, System specifications documents, documents hierarchies**

I. Introduction

System specifications are generally organized according to several documents hierarchies. These hierarchies levels are linked and represent the traceability information. This latter is used for example in order to verify the coverage analysis, to the reuse of product components and for project status analysis.

In requirement engineering (RE) context, industrial requirements should be high quality documents. This means that they should respect some properties that guarantee an obtaining of the wanted final product. Several quality properties are defined in the literature ([1] [2]). We are interested in "linked Set" defined in [1] as an "*explicit relationships should be defined among individual requirements to show how the requirements are related to form a complete system*".

Several definitions are proposed for traceability in requirement engineering context. Traceability is defined according to [1] as: "*The degree to which can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; e.g., the degree to which the requirements and design of a given system element match. (IEEE Std 610.12-1990)*". According to [3], "*requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction*".

Traceability is done through different specifications levels. High-level specifications aim to understand the objectives, goals, aims, aspirations, expectations and needs in order to transform them into low-level specifications (i.e. components, materials, etc.). Figure 1 gives some examples of different specifications levels used in RE context.

In RE context, the costs to fix errors increase much more after that the product is built than it would if the requirements defects were discovered during the requirements writing phase of a project [4] [5] [6]. That's why, when writing or revising a set of requirements, or any technical document, it is particularly challenging to make sure that texts are correctly and completely linked for any domain actor. Manually identifying linked requirements is an obviously time-consuming and costly task. Also, it needs RE experts to establish link according to their domain-based knowledge. We tackle this problem in term of similarity between requirements from different specifications levels.

We focus in this paper on how to generate automatically linked requirements through different levels in order to guarantee the coherence of the linked set of specifications. Our approach allows then to save time which contribute to reduce the project cost. The two main scenarios that we can use this method for are: 1. Test the quality of the generated links after the generation of the traceability matrix (TM). In fact, in order to control the completeness of the TM, it is useful to compare the generated TM by experts with the generated TM by our approach. 2. Our client has a lot of documents from an archived project and he wants to generate automatically the TM in order to avoid manual processing of this task. So, our approach is useful for this task since it is less time-consuming.

The problems of traceability managing can be handled according to different technologies. We focus on artificial intelligence (AI) approaches and more precisely classification approaches. Automatic classification of requirements is widely studied in the literature using: convolutional neural networks [7], naives bayes classifier [8], text classification algorithms [9]. Data classification approaches could be data clustering through algorithm such as k-means. This latter is studied in different contexts due to its efficiency [10].

This paper is a continuity of our previous paper [11], dealing with requirements quality in terms of redundancy and inconsistency. So, the main contribution of this paper is to adapt the use of k-means algorithm for a traceability managing

DOI 10.1109/RE.2019.00035

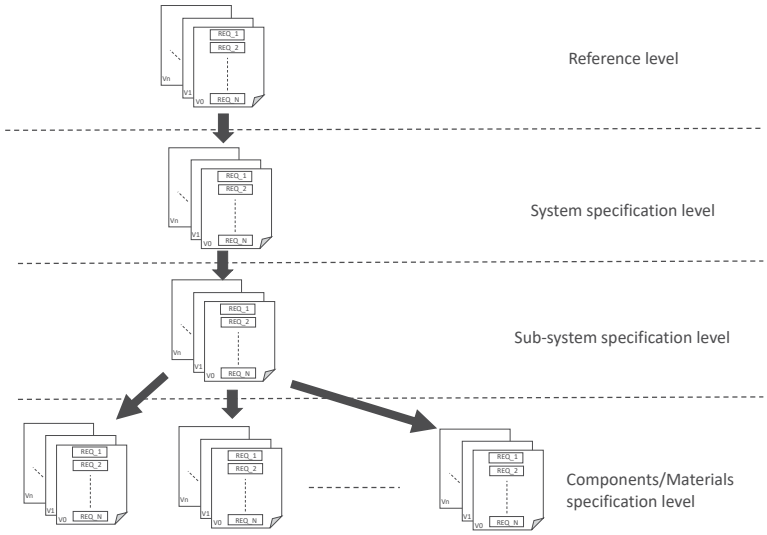

Fig. 1. An example of different specifications levels

in RE context. In fact, instead of applying k-means to a single document, we will apply it to a couple of linked documents.

This paper is structured as follows: in section II, we present related works on the traceability managing through artificial intelligence approaches by focusing on the k-means technique. In section III, we present our traceability management approach and we explain the validation approach used to evaluate the relevance of our results. In section IV, we present the datasets used to evaluate our approach and the results obtained by applying our clustering approach. We highlight also the impact of the pre-processing treatment on the results. In section V, we discuss the associated results. In section VI, we conclude and give some future research directions.

II. Related works

In this section, we first present related works associated to traceability tools used in specifications documents or technical documents. Second, we present how traceability information could be visualized. Finally, we focus on AI based approach such as k-means for clustering and how we could exploit this algorithm for a traceability purpose.

A. Traceability tools

Manual traceability. Traceability is realized by capturing traces either entirely manual or tool supported, e.g. as spreadsheet in Microsoft Excel. Traceability remains a challenge since it is cumbersome, error-prone, and often leads to traceability information that is of poor quality due to the very high number of artifacts to be traced and the quantity of involved development tools [12].

Tool-supported traceability. The following approaches exist to develop homogenized and aggregated information that is distributed across a whole chain of development tools:

- Homogenization of the tool environment through an ALM tool (Application Lifecycle Management). ALM tool chains cover the whole life-cycle of a system and manage all artifacts of the development process in a holistic approach. The advantage of this approach is that the homogenization of artifacts allows managing and analyzing them easily with dedicated tools of the ALM tool. The disadvantage is that it is necessary to implement the whole ALM tool chain. If introduced, it is difficult to replace specific tools in the tool chain.
- Homogenization of data through surrogate requirements. Requirements management (RM) tools allow storing, organizing, and managing all requirements of a system's specifications and typically arrange them in a specification tree that links each requirement to its parent requirement in the higher specification. Commercial RM tools with traceability support are, e.g., IBM Rational RequisitePro, IBM DOORS and Visure Requirements. Typical analysis functions based on recorded traceability information are e.g., completeness checks, and assessment of requirements deviations over all levels. The disadvantage of this approach is that different adapters or converters for the different artefact types are necessary that need to have a consistent version and data format. In contrast to ALM tools, this consistency must be carried out oneself.

B. Visualization of traceability information

Visualization of traceability information aims to help the users to describe and to track the relationships between different software artifacts. Several techniques of traceability exist and are used in different contexts. Common visualizations for traceability information are matrices, graphs, lists, and hyperlinks. We explain briefly each technique:

- **Matrix:** A traceability matrix (TM), usually in the form of a table, is used to assist in determining the complete-

ness of a relationship by correlating any two documents using a many-to-many relationship comparison [13]. It is often used with high-level requirements and detailed requirements of the product to the matching parts of high-level design, detailed design, test plan, and test cases. According to [13], the advantage of traceability matrices is that all links between artifacts are visible at a glance. Filters help to reduce the amount of displayed information. Traceability matrices are suitable for management tasks. However, in industry, projects often consist of thousands of artifacts: the tables could become very large and confusing.

- **Graph:** In a traceability graph, artifacts are represented as nodes connected by edges, if a trace link between the artifacts exists. It allows getting an overview on the links and then gives a high information comprehension ratio. Navigating through the graph makes easy to identify missing links [13].
- **List:** Represent traceability links in one entry. This entry could include information concerning the source and target artefact and attributes. They are especially suitable when bulk operations for several different artifacts should be executed. Filters and sorting mechanisms allow to handle the displayed information. However, compared to the visualizations described above lists are less suitable to execute project management, development and testing tasks [14].
- **Hyperlinks:** Hyperlinks connect linked artifacts and allow jumping from a source artefact to a linked artefact. This visualization is suitable if detailed information about an artefact is needed as it allows navigation to artifacts in their native environment. According to [14] using hyperlinks solely has the disadvantage that a lot of navigation effort is necessary to get an overview on the link status as linked artifacts are not visualized compactly.

According to [14] "*traceability matrices and graphs are most preferred in management tasks, while hyperlinks are preferred in implementation and testing tasks. Traceability lists seem to be the least attractive technique for most participants. Graphs are preferred to navigate linked artifacts, while matrices are appropriate for overview. Hyperlinks are regarded to fit for fine-grained information.*". We focus in this paper on analysing matrices, since they are useful for management tasks.

C. Artificial intelligence for traceability

Artificial intelligence is a challenging approach to deal with traceability problem. In fact, linked technical documents need a certain domain expertise and knowledge to figure out which requirement is linked to which other requirement. We focus on link generation and we treat it as a similarity problem. In fact, two linked specifications are somehow similar. In order to group similar specifications, we can use a clustering approach. The notion of a "cluster" remain not clear, and that's why there are so many clustering algorithms. Each algorithm is based on some characteristics that define the clusters model such as:

- Connectivity models: based on distance connectivity e.g. hierarchical clustering.
- Centroid models: represents each cluster by a single mean vector e.g. k-means algorithm.
- Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the expectation-maximization algorithm.
- Density models: for example, DBSCAN defines clusters as connected dense regions in the data space.
- Subspace models: in bi-clustering (also known as co-clustering or two-mode-clustering), clusters are modeled with both cluster members and relevant attributes.
- Graph-based models: based on clique (a subset of nodes in a graph) such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster.
- Neural models: the most well known unsupervised neural network is the self-organizing map (SOP) and these models can usually be characterized as similar to one or more of the above models, and including subspace models when neural networks implement a form of Principal Component Analysis or Independent Component Analysis.

For a traceability purpose, we use a centroid model which is an unsupervised machine learning for clustering (k-means) due to its popularity and the the rapidity especially for large datasets. In fact, popularity is a good metric that reflect the efficiency of this model. Rapidity is useful to treat large specification document. The choice of k-means is also motivated by the fact that it is language independent since it is statistical model. So, we can apply it on specification issued from different languages.

K-means clustering aims to partition n observations (requirements in our case) into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. We presented in our previous work [11] a detailed explanation of how k-means works from a mathematical point of view.

III. Traceability Approach

As cited above, this approach is inspired from our previous approach of clustering to detect redundancy and inconsistency in industrial requirements [11]. We focus in this paper on analyzing linked system specifications and how we can use k-means algorithm in order to generate the appropriate links.

The main steps of our approach are shown in Figure 2. The novelty of this approach compared to our previous work [11] is that we analyse two related documents instead of one. Given an industrial specification, we extract first the requirement files from each specification[1] containing only requirements to analyze using a predefined function in *SEMIOS for Requirements*[2] software. We merge each specification with its linked upper-level specification and analyse them as one file. Second,

[1]Note that we analyse only textual specification (.txt, .doc, .docx)
[2]http://www.semiosapp.com/

we detect only duplicates requirements belonging to distinct documents. Third, we apply a k-means clustering algorithm on the non duplicated requirements. Last, we merge duplicated requirements and clusters results in order to obtain our final results. We detail and explain these steps in the sections below.

We explain also in this section, the validation approach in order to evaluate the obtained results.

A. Requirements extraction : SEMIOS for Requirements

SEMIOS for Requirements is a proofreader tool for specifications from the conception phase. The core of the semantic engine of this tool is based on NLP techniques and works directly with RE domains tools like IBM DOORS, IBM Doors Next Generation, MS Word, MS excel, etc. It aims to control specifications quality and reduce management cost. Requirements extraction is based on a predefined regular expression (i.e. a pattern) and/or a predefined Microsoft Word Style. This allows us to detect the beginning and the end of each requirement. The traceability approach that we propose in this work will allow us to add a new functionality to *SEMIOS for Requirements*.

B. Duplicates detection

Duplicates are easily detectable through a simple comparison of two requirements. However, their impact is very important. In fact, detecting and then discarding duplicates from the requirements set in the next step (clustering) will help us to detect only related information. Also, detecting duplicates (belonging to different documents) is essential to build the traceability matrix since two requirements could be linked in the TM. So, instead of doing it manually (by the RE engineers), this step is automatically done via a string matching. This detection contribute to have part of the results in the TM.

C. K-means algorithm

Since the k-means algorithm already detailed and explained in our previous work [11], we present in this paper only how we use it in a multi-document context.

K-means aims to group/cluster requirements according to their similarity. K-means is an unsupervised machine learning approach, which means that we don't need any labeled data to perform clustering. This is very useful in RE context, since we can apply it to different domains without any prior knowledge. Also, k-means is a statistical model and then language independent. Our approach is a generic approach which deals with any domain and any language.

In our work, k-means is used to group "similar" requirements issued from different specifications. To do that, we merge each two specifications from different levels together and we apply k-means (according to some predefined criteria already discussed in our previous work [11]). Once we have the clustering results, we discard clusters with only one requirement (since there is no link). Also we discard clusters with requirements belonging to the same document (we are interested to find the relation between specifications from different requirements). As a final result, we will have only clusters that reflect linked data from distinct documents.

In order to illustrate the results of the k-means algorithm, we give an example of clustering results. Let's assume that we have a list of 8 requirements extracted from two specifications as follows :

- From document 1:

1) *When PACK shut off sequence is activated, the APU flow demand shall be forced to 0%.*
2) *During ACU starting sequence, on APU, the APU Flow demand shall be driven to 100%*
3) *The Flow Control Valve (FCV) shall be a pneumatically actuated, electrically controlled, butterfly valve.*
4) *The maximum external leakage, valve open or closed, shall be less than 2 g/s under 2.5 bar gauge upstream pressure at room temperature, sea level.*

- From document 2:

1) *When the Flow Control Sequence is "CLOSING" AND APU activation is "active", the APU flow demand shall be forced to 0%.*
2) *When the Flow Control Sequence is "STARTING" AND APU activation is "active", the APU flow demand shall be set to 100%*
3) *The FCV shall be a pneumatically actuated, electrically controlled butterfly valve.*
4) *The external leakage of the TAPRV actuator, in open position, shall not exceed 2 g/s with upstream butterfly pressure of 4.1 barg at 20C and sea level.*

K-means algorithm will cluster this list into a set of *k* fixed number of clusters. Let's assume that k=4, the result of the algorithm will be as follows:

- Cluster 1:
  1) *When PACK shut off sequence is activated, the APU flow demand shall be forced to 0%.*
  2) *When the Flow Control Sequence is "CLOSING" AND APU activation is "active", the APU flow demand shall be forced to 0%.*
- Cluster 2:
  1) *During ACU starting sequence, on APU, the APU Flow demand shall be driven to 100%*
  2) *When the Flow Control Sequence is "STARTING" AND APU activation is "active", the APU flow demand shall be set to 100%*
- Cluster 3:
  1) *The Flow Control Valve (FCV) shall be a pneumatically actuated, electrically controlled, butterfly valve.*
  2) *The FCV shall be a pneumatically actuated, electrically controlled butterfly valve.*
- Cluster 4:
  1) *The maximum external leakage, valve open or closed, shall be less than 2 g/s under 2.5 bar gauge upstream pressure at room temperature, sea level.*
  2) *The external leakage of the TAPRV actuator, in open position, shall not exceed 2 g/s with upstream butterfly pressure of 4.1 barg at 20C and sea level.*

The algorithm clusters the requirements according to their similarities. So, each cluster contains the most similar requirements. A cluster may contain one or more requirements depending on the dataset. We have shown in this example only the case of 2 requirements per cluster.

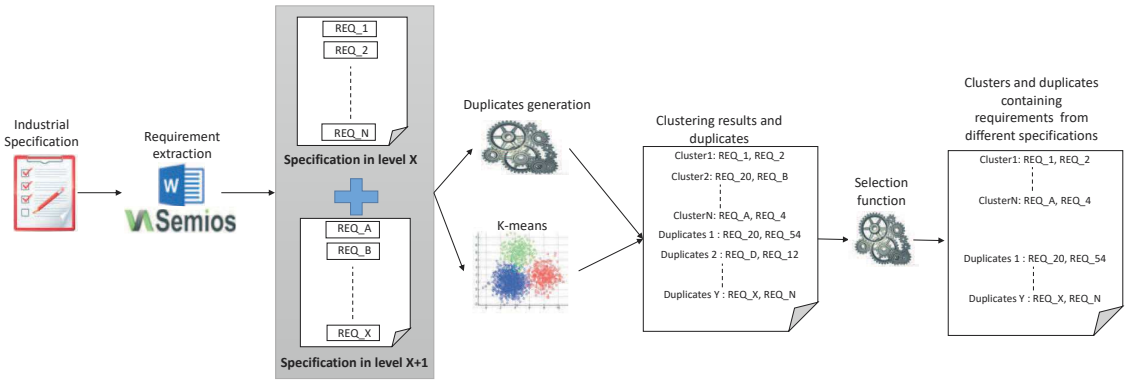

Fig. 2. Traceability approach overview

D. Validation approach

Since the proposed approach is based on an unsupervised machine learning algorithm, we do not have any labeled data to validate our results. So, we will use the traceability matrix already given by RE engineers, as a ground truth in order to validate our results. In fact, considering two specifications from different levels, we can compare our clustering results according to the information provided by this matrix. This process is explained in figure 3.

We evaluate our approach according to the precision of the clusters (how precise/accurate our model is out of those predicted positive, how many of them are actual positive).

IV. Experimentation Results

In this section, we present in section IV-A the datasets used in the experiment. In section IV-B, we present the result of our approach. In section IV-C, we present the results obtained with pre-processing step.

A. Datasets

In order to test our approach, we extracted a list of requirements from 2 industrial datasets (written in English). For confidentiality issues, we are not allowed to reveal the identity of the companies. These datasets contain different levels of specifications. We present characteristics of these datasets as follows:

- **Dataset1:** Belongs to the space domain. It contains two levels of specification. Level L0 is the highest level (client specification) and contains 762 distinct requirements. Level L1 is the second level and contains 521 distinct requirement. In this dataset, we will analyse one traceability matrix (between L0 and L1). In this latter, each requirement is related to only one other requirement.
- **Dataset2:** Belongs to the aeronautic domain. It contains five levels of specifications, 14 specifications documents and 15 relations between specifications. We present Figure 4 in order to understand the different existing links. The specifications in each level that we will analyse are mentioned with the associated number of requirements as follows:
  - Level 0: L0 (190), L10 (559),
  - Level 1: L11 (203), L12 (561), L13 (538), L14 (288),
  - Level 2: L20 (50), L21 (1586), L22 (163), L23 (51), L24 (72),
  - Level 3: L30 (40), L31 (34), L32 (43).

For each corpus, we have the upper (linking the high level specification with a lower level specification) and the lower (linking the low level specification with a higher level specification) traceability matrix. This matrix may contain two relations type: 1) one to one relation type (each requirement is related to only one requirement) for example, Dataset1; 2) one to many relation type (each requirement is related to one or many requirements) for example, Dataset2.

B. Results of our approach

We have tested our approach on each dataset presented in section IV-A. We remind that our purpose is to generate the traceability matrix automatically. We calculate the number of relevant clusters (they contain linked requirements grouped into clusters) according to a given traceability matrix generated by the RE expert. We present the results associated to the precision (P.) in Table I. The results are obtained with value of k equal to the total number of requirements - 20%. For example, if we have 100 requirements, k is equal to 80. We will generate 80 clusters and we will write only clusters with a number of requirements more than 1 and also with requirements belonging to distinct specifications.

In Table I, we calculate two precision values: 1. the precision of the written clusters that reflects the percentage of relevant clusters found in the clustering results, 2. the precision according to the traceability matrix (TM) that reflects the percentage of relevant clusters found comparing to those in the TM.

We highlight that there are many requirements which contain only the word ”Deleted”. These requirements are written in the duplicates part. Since they are not related to other requirements, we discard these requirements from our precision calculation.

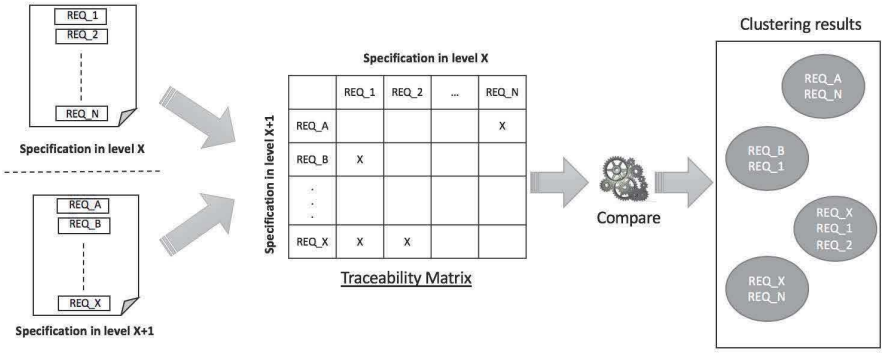

Fig. 3. Traceability validation overview

TABLE I
RESULTS OF THE TRACEABILITY APPROACH ACCORDING TO K-MEANS ALGORITHM

Analysed spec.	nb. req.	nb. dup	nb. analysed req	K value	nb. linked req in TM	nb. written clusters	P. of clusters	P. % to the TM
Dataset1: LO-L1	1283	257	1026	820	367	88	84%	75.88%
Dataset2: LO-L10	749	282	467	373	33	5	20%	15%
Dataset2: LO-L12	751	246	505	404	32	5	40%	12.5%
Dataset2: LO-L13	728	97	631	504	62	8	12.5%	4.88%
Dataset2: LO-L14	478	9	469	375	5	4	0%	40%
Dataset2: LO-L1	2339	641	1698	1358	73	5	40%	13.69%
Dataset2: L11-L20	252	71	182	145	23	8	50%	21.74%
Dataset2: L11-L21	1789	89	1700	1360	29	0	0%	13,79%
Dataset2: L12-L21	2147	265	1882	1505	71	0	0%	5.6%
Dataset2: L13-L21	2124	116	2008	1606	115	0	0%	2.61%
Dataset2: L14-L21	1874	31	1843	1474	84	1	0%	5.95%
Dataset2: L13-L22	701	133	568	454	40	5	71.42%	30.25%
Dataset2: L13-L23	588	95	493	394	21	0	0%	0%
Dataset2: L13-L24	610	96	514	411	19	0	0%	0%
Dataset2: L13-(L21+L22+L23+L24)	2409	157	2252	1801	80	1	100%	18.75%
Dataset2: L30-L22	203	33	170	136	40	2	100%	15%
Dataset2: L31-L22	197	34	163	130	15	0	0%	33.33%
Dataset2: L32-L22	206	33	173	138	39	2	100%	15.38%
Dataset2: L32-L30	83	0	83	66	2	14	14.28%	100%
Dataset2: L22-(L30+L31+L32)	280	35	245	196	34	2	100%	23.52%

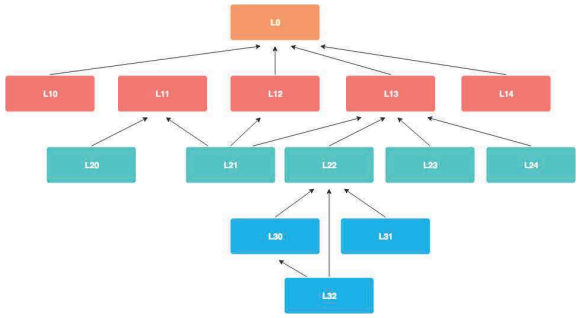

Fig. 4. Dataset2 : levels and links

According to the results shown in Table I, we obtain good results for **Dataset1**. This is explained by the fact that: 1) we have one requirement linked to only one other requirement, 2) a lot of requirements are written in similar syntax, and 3) the requirements are written with a few acronyms, abbreviations and technical terms. The characteristic of this dataset is suitable for our approach.

According to the results shown in Table I, we detail the results for **Dataset2**:

- For the results between the level L0 an L1, the precision of the clusters and the precision according to the TM do not exceed 40%. L0 is the highest level of the hierarchy and represent the client specification. It is then written with long phrases and with details associated to the client need. L1 is the lower level of L0. So, it contains more technical terms, it is more precise and more concise than L1. Our approach can partially handle these characteristics.
- For the results between the level L1 and L2, the precision of the clusters is 0% except for the L13-L23 (71.42%) and L13-(L21+L22+L23+L24) (100%). These two levels contain

a lot of technical and domain related terms. The requirements aren't written as long phrases, but as short ones. Also, the links between the requirements need an expert's knowledge to be established.

- For *Dataset2: L32-L30*, the number of written clusters is higher than the number of linked requirements in the matrix. The majority of the written clusters are very similar in terms of syntax, however they are not stated as linked in the matrix. For example, this cluster is stated as linked in the clustering results but does not figure in the matrix:

- *Operational Shocks and Crash Safety, XX 7, category B*
- *Operational shocks and crash safety XX 7 B*

- For the results between the level L2 and L3, they are better than the results between L1 and L2. In fact, in L3 we found technical terms but used in long phrases and a few formulas and symbols.

Concerning the number of written clusters in **Dataset1** and **Dataset2**, we can see that our approach detect few clusters comparing to the linked requirements in the TM. However, in Dataset1, we can notice that we have a lot of written clusters. Our approach is then more suitable for one to one relation type than one to many relation type.

The overall of the experiments shows that this clustering approach provide better results for high level requirements which are written with few technical terms, acronyms, abbreviations, formulas and symbols. Requirements that need an expert knowledge are not well detected.

In order to improve the efficiency of our approach in the lowest levels, we proceed to a pre-processing step explained in the next section.

C. Results of our approach: with technical terms detection and syntactic pre-processing

Since the lowest levels of a specification contain more technical terms, we should take them into consideration in our analysis. So, before applying the k-means algorithm, we proceed to a pre-processing as follows:

- Technical terms detection: according to postags patterns defined by RE experts (already detailed in our previous works [11]). For example, the detected technical terms are *water extractor temperature*, *system operating range conditions*, *General Technical Specification* and *Power Input*. These terms are stated in our analysis as one word instead of several words. This allows us to give more weight to each group of technical terms instead of considering each one as a single piece of data.
- Syntactic pre-porcessing: lemmatization and stemming are text normalization techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing. Stemming and lemmatization helps us to achieve the root forms.

In Table II, we present the obtained results by applying this pre-processing step.

These results show a significant improvement in terms of the numbers of written clusters. Also, we can notice a better precision value of the written clusters and the precision according to the TM in most of the levels. technical terms detection and syntactic pre-processing homogenize better requirements making clustering more efficient.

V. Discussion

As discussed previously, the level of the specifications impact the precision results of our approach.

We compare the precision of the generated clusters with and without the pre-processing step in Figure 5. We can notice that the pre-processing is useful for the high level since we obtain better results. Precision without pre-processing is better for the lowest level. This is explained by the fact that we have more written clusters with the pre-processing, so the precision may decrease if these clusters aren't relevant.

In Figure 6, we compare the precision of the clusters comparing to the TM with and without the pre-processing step. We can notice that the pre-processing barely improved the results for the whole levels. These minimal differences between the two curves are explained by the fact that the TM is generated by experts and then we need more than just a pre-processing step to achieve better results.

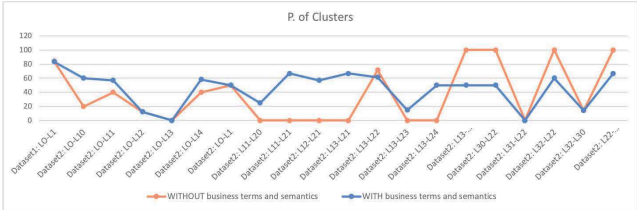

Fig. 5. Comparison of the precision values of the generated clusters with and without the pre-processing step

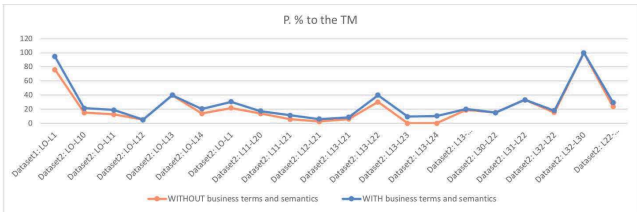

Fig. 6. Comparison of the precision values of the clusters in the TM with and without the pre-processing step

Taking into consideration technical terms and syntactic improve the average precision of our approach: from 36.61% to 45.20% for the average precision of the written clusters and from 22.39% to 27.45% for the average precision related to the TM. The detection of technical terms is pattern-based multi-word detection so it is domain independent. So, we can use this pre-processing on different specifications from different domains.

Our approach could be integrated in current traceability management systems. In fact, practitioners could use our approach in order to generate the TM or a part of it. It could

TABLE II
RESULTS OF THE TRACEABILITY APPROACH ACCORDING TO K-MEANS ALGORITHM: WITH TECHNICAL TERMS AND SYNTACTIC PRE-PROCESSING

Analysed spec.	nb. req.	nb. dup	nb. analysed req	K value	nb. linked req in TM	nb. written clusters	P. of clusters	P. % to the TM
Dataset1: LO-L1	1283	257	1026	820	367	90	83.33%	94.55%
Dataset2: LO-L10	749	282	467	373	33	10	60%	21.21%
Dataset2: LO-L12	751	246	505	404	32	7	57.14%	18.75%
Dataset2: LO-L13	728	97	631	504	62	8	12.5%	4.88%
Dataset2: LO-L14	478	9	469	375	5	4	0%	40%
Dataset2: LO-L1	**2339**	641	1698	1358	73	12	58.33%	20.54%
Dataset2: L11-L20	252	71	182	145	23	12	50%	30.43%
Dataset2: L11-L21	1789	89	1700	1360	29	4	25%	17,24%
Dataset2: L12-L21	2147	265	1882	1505	71	6	66.66%	11.26%
Dataset2: L13-L21	2124	116	2008	1606	115	7	57%	6.08%
Dataset2: L14-L21	1874	31	1843	1474	84	3	66.66%	8.33%
Dataset2: L13-L22	701	133	568	454	40	13	61.53%	40%
Dataset2: L13-L23	588	95	493	394	21	3	15%	9.52%
Dataset2: L13-L24	610	96	514	411	19	4	50%	10.52%
Dataset2: L13-(L21+L22+L23+L24)	**2409**	157	2252	1801	80	4	50%	20%
Dataset2: L30-L22	203	33	170	136	40	4	50%	15%
Dataset2: L31-L22	197	34	163	130	15	0	0%	33.33%
Dataset2: L32-L22	206	33	173	138	39	5	60%	17.94%
Dataset2: L32-L30	83	0	83	66	2	14	14.28%	100%
Dataset2: L22-(L30+L31+L32)	**280**	35	245	196	34	6	66.66%	29.41%

be also used in order to check the relevance of an existent TM.

VI. CONCLUSION

In this paper, we have proposed a clustering-based approach for traceability management. We have analysed specifications from different levels. Our approach provides better results when the requirements are written in natural language (i.e. high level specifications). The pre-processing step improved the precision results for the whole levels. This approach is language independent since k-means is a statistic algorithm, and domain independent since technical terms detection is pattern-based multi-word detection.

The main difficulty associated to RE context, is the expert knowledge that couldn't be totally managed by our approach. In order to have a better knowledge of the domain, we plan to capture the context of each specification through a neural network approach, before applying our clustering algorithm. However, we need a lot of data to train an efficient model. We plan to observe how the K-means algorithm performs when applied to a data set where other AI traceability techniques are applied.

ACKNOWLEDGEMENTS

We would like to thank Audrey Speronel from "OneLight Studio" for her contribution in our experiments.

REFERENCES

[1] IEEE, "IEEE guide for developing system requirements specifications," *IEEE Std 1233, 1998 Edition*, 1998.

[2] R. INCOSE, "Guide for writing requirements," *Version 2. Prepared by: Requirements Working Group*, 2015.

[3] O. C. Z. Gotel and A. C. W. Finkelstein, "An analysis of the requirements traceability problem," 1994, pp. 94–101.

[4] R. L. Glas, *Facts and Fallacies of Software Engineering*. Addison-Wesley Professional, 2002.

[5] B. D. B. H. R. L. Jonette M. Stecklein, Jim Dabney and G. Moroney, "Error cost escalation through the project life cycle," in *Proceedings of the 14th Annual International Symposium*, Toulouse, France, 2004.

[6] A. A. Alshazly, A. M. Elfatatry, and M. S. Abougabal, "Detecting defects in software requirements specification," *Alexandria Engineering Journal*, vol. 53, no. 3, pp. 513 – 527, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1110016814000568

[7] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, Sept 2016, pp. 39–45.

[8] E. Knauss, D. Damian, G. Poo-Caamao, and J. Cleland-Huang, "Detecting and classifying patterns of requirements clarifications," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, Sept 2012, pp. 251–260.

[9] D. Ott, *Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 50–64. [Online]. Available: https://doi.org/10.1007/978-3-642-37422-7_4

[10] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010, award winning papers from the 19th International Conference on Pattern Recognition (ICPR). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865509002323

[11] M. Mezghani, J. Kang, and F. Sèdes, "Industrial requirements classification for redundancy and inconsistency detection in SEMIOS," in *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, 2018, pp. 297–303. [Online]. Available: https://doi.org/10.1109/RE.2018.00037

[12] A. Kannenberg and H. Saiedian, "Why software requirements traceability remains a challenge," *The Journal of Defense Software Engineering*, vol. 22, pp. 14–19, 07 2009.

[13] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. Maletic, and P. Mäder, *Traceability Fundamentals*. London: Springer London, 2012, pp. 3–22. [Online]. Available: https://doi.org/10.1007/978-1-4471-2239-5_1

[14] Y. Li and W. Maalej, "Which traceability visualization is suitable in this context? a comparative study," in *Requirements Engineering: Foundation for Software Quality*, B. Regnell and D. Damian, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 194–210.